

AD-A151 992

AN IKBS APPROACH TO IMAGE UNDERSTANDING(U) ROYAL
SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND)

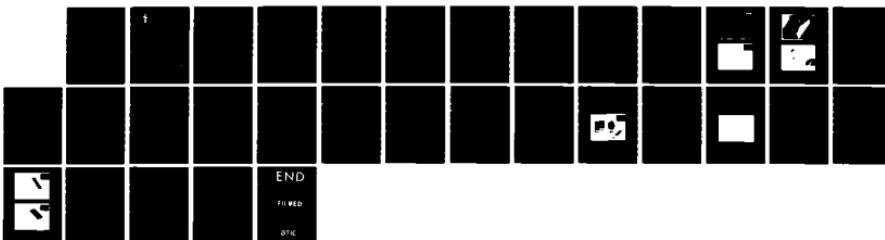
1/1

A C SLEIGH ET AL. OCT 84 RSRE-MEMO-3683 DRIC-BR-94355

F/G 12/1

UNCLASSIFIED

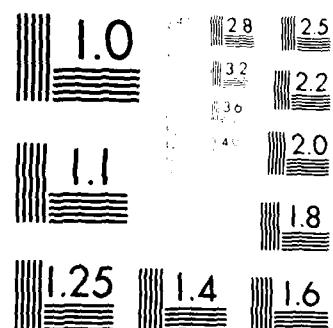
NL



END

FILED

DTIC



MICROCOPY RESOLUTION TEST CHART
Nikon Model 1200 Series

(2)



AD-A151 992

RSRE
MEMORANDUM No. 3683

**ROYAL SIGNALS & RADAR
ESTABLISHMENT**

RSRE MEMORANDUM No. 3683

DTIC FILE COPY

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

"Original contains color plates: All DTIC reproductions will be in black and white."

SDTIC
SELECTED
MAR 28 1985

UNLIMITED

85

3 27 005

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3683

Title: AN IKBS APPROACH TO IMAGE UNDERSTANDING

Authors: A C Sleigh and D B Hearn

Date: October 1984

SUMMARY

This memorandum describes a method for extracting 2-Dimensional and 3-Dimensional shapes from scenes made up entirely of straight line segments. A new approach to extracting boundaries is discussed, and a methodology for defining geometrical shapes in PROLOG illustrated. Emphasis is placed on the structuring and control strategy for efficient implementation, and the applicability of IKBS methods to this is demonstrated.

This memorandum is for advance information. It is not necessarily to be regarded as a final or official statement by Procurement Executive, Ministry of Defence

Copyright
C
Controller HMSO London

1984

AN IEBS APPROACH TO IMAGE UNDERSTANDING

A C Sleigh and D B Hearn

July 1984

CONTENTS

	Page
<u>1. INTRODUCTION</u>	2
<u>2. DETAILS OF MODEL FORMING STAGES</u>	5
2.1 Finding Boundaries	5
2.2 Shape and Object Finding	11
<u>3. CONCLUSION</u>	27

...and the other two, the *littera* and *littera* *ad litteras*, are the *littera* *ad litteras* and *littera* *ad litteras*.

1. INTRODUCTION

Image understanding is concerned with the important area of interpreting the output of a sensor and initiating an appropriate response to the perceived scenario. This might mean rejecting a component on a production line because of some defect, selecting the highest value target accessible to a missile, or sending a meaningful message down a narrow bandwidth channel. Current device and software technology will enable very high-performance computers to have widespread use in a vast range of new products. In most cases this computing power will need to interact directly with the environment rather than through a human input, and image understanding techniques provide the mechanism to bypass this human bottleneck. These techniques must be capable of representing a large body of knowledge and experience in order to cope with the variability and complexity of unconstrained environments.

Even in fairly simple situations, the combinatorial implications can be surprisingly demanding: a typical TV frame can represent over $10^{100,000,000}$ distinct patterns, and a sketch of 100 lines can be linked to form millions of potential polygons. Not only is it necessary to define shapes or objects in a unique way, it is also necessary to find a way of analysing image data which avoids the need to consider all possible outcomes, or even a very small fraction thereof. The methods associated with the labels "IKBS" or "AI" have much to contribute to this aspect of the problem, for as well as providing a convenient formalism for representing knowledge, they provide methods which allow definition of processes which manipulate this knowledge, form problem solving strategies, and give the programmer control at progressively higher orders of expertise.

The ability to easily define the problem domain itself, and also to define higher domains concerned with representing problem solving knowledge, clearly separate AI methods from those associated with statistical classification or associative memories. Statistical classifiers can extract relationships between data items and can therefore deduce knowledge about the problem domain, but they cannot exploit higher level knowledge, such as generalisation or analogy. The ability to form a "system of knowledge" using higher level processes is critical for vision systems. Without it vision is an ill-formed problem: the dimensionality of the problem space usually far exceeds that of any conceivable training set.

The image understanding approach described in this paper forms such a "system of knowledge" by a hierarchy of model forming stages, each level using methods which have strong IKBS/AI foundations. This hierarchy (Figure 1) serves a number of purposes:

- It enforces an order into the way image data is analysed which maps onto the way we want to describe objects. This reduces the

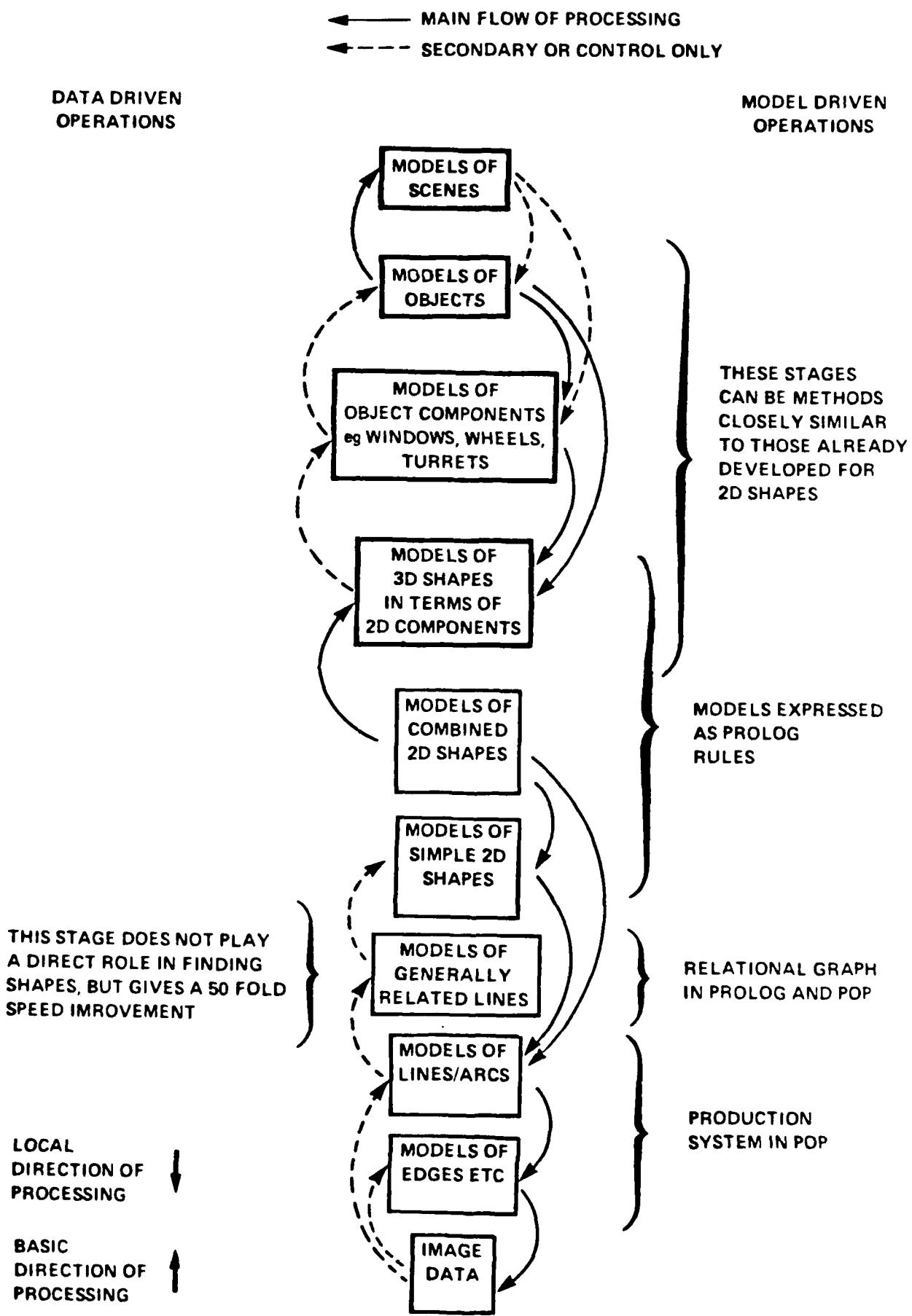


Figure 1. Hierarchy of Models Used to Interpret Sensor Output

combinatorial factor and simplifies the addition of new expertise.

- It allows knowledge to be separated into independent databases.
- It allows different problem solving (and learning) methods to be used in different places.
- Each model forming stage is a candidate for a separate processor in a concurrent implementation.

The flow of information between the model forming stages of Figure 1 is quite complex, and makes use of both forward data driven processes and backward goal driven process as indicated by the arcs between the boxes. In general, data driven processes are used to form cues which guide a goal driven process, the latter playing the main role in inferring the outcome. The forward processes do not necessarily influence the outcome, but play an important role in determining how quickly the right model is applied. The goal directed path is used to carry out the main part of the inference operation because of the nature of image description. The number of possible goals is usually very much smaller than the amount of input data, a situation which favours backward reasoning, especially as the input data may have many distinct interpretations depending upon the context. Without a model to impose suitable contexts the problem explodes combinatorially; with the right choice of model it is tractable. Our choice of models, and therefore the detail of the boxes in Figure 1, has been driven by the need to avoid combinatorial instability. In developing this system we have frequently encountered combinatorial problems, but we have always been able to re-define the hierarchy to overcome these [There may be a formal way of identifying and eliminating combinatorial instability, but we have not yet tackled this interesting piece of theory].

The hierarchy of models shown in Figure 1 starts with the image input which is analysed into line segments using a two stage model, one is concerned with the conditions which must be satisfied if a line is to be extended to include a particular point, and one with determining the local properties around points which are candidates for extension. These processes are driven by backward inference from the line model, but there is a controlling forward route to guide the line model's operation. The line segments are then analysed into straight line segments and arcs (arcs are not yet implemented), which are expressed as a PROLOG database for the subsequent analysis by an expert system concerned with the identification of two dimensional polygonal shapes. Again this is done by a model driven process, but with a strong forward control method which, although not participating directly in the solution, does enable the correct model to be applied approximately 50 times more quickly. Note that all polygonal shapes present are identified, individual lines being used any number of times to form different shapes. The two dimensional shapes so found are analysed further to assert other latent shapes. For example, a rectangle with one corner obscured can be identified by a rule concerned with shapes having three right angles, and shapes which are split can be re-joined. Finally, two dimensional shapes which meet in a way consistent with a three dimensional interpretation are found and the positions of the vertices identified. We now have a database which consists of three dimensional shapes, two dimensional shapes, and the underlying boundary segments. These form the input to the highest stage in the image

understanding process where candidate objects are tried for consistency with this database. In some cases the database will directly define the object, in others it will only suggest a range of possible objects and orientations to be examined in detail using the boundary database. This part of the process has yet to be developed, but it is intended to use a three dimensional matching process similar to that of Hogg (Sussex) and Sullivan (Plymouth Polytechnic) to test for the existence of objects which have been cued in the way described here.

2. DETAILS OF THE MODEL FORMING STAGES

2.1 Finding Boundaries.

Images are made up of regions of differing intensity and texture. The location of the boundaries between these regions contains most of the information in an image, as illustrated by artists such as Aubrey Beardsley. Finding these boundaries is not an easy process. This is because although boundaries have a few features in common, the detail of their makeup is extremely variable. In addition to noise and blurring, edges can be sharp or gradual, they can be straight or wavy, demark two regions of similar texture but different intensity, or areas of different texture but identical mean brightness. The complexity of boundaries have forced some workers to suggest that low level boundary extraction is an impossible task, and that a high level model must be imposed before any interpretation can be formed. Others have largely ignored the richness of boundaries and relied on simple gradient operators (eg Sobel, Marr-Hildreth) which in-accurately capture only a subset of boundary types, in the hope that the confusion caused by missing or extraneous lines can be removed by higher level operations. In contrast to both these approaches, we have attempted to develop methods which can distinguish the variety of boundary types without any high level input, passing a boundary description to the higher level processes to use as they wish. The key philosophy here is to avoid introducing high level information at the pixel level (because of combinatorial implications), but instead to use a large amount of information local to the task of robustly identifying boundaries.

It is possible to find and discriminate boundaries by a number of methods, including direct pattern matching, statistical classifiers and rule based descriptions. Pattern matching requires an inconveniently large number of templates because of the extreme variability of boundary types. Statistical classifiers are a feasible candidate, but the choice of appropriate features is made difficult by the differing scales, texture and sharpness of boundaries. In comparison, a rule based approach offers several advantages. It is possible to specify the particular features which discriminate one type of boundary from an other in a form which has a direct relationship to our (human) understanding of what demarks a boundary. As in the description of plant or animal types, where particular attributes, such as the number of points on a leaf, are used to distinguish a species, it is possible to construct a system of description which interrogates only the critical features to identify the properties within a portion of the picture. Not only does this

provide a suitable way of entering knowledge into the system, it can reduce the number of features which must be evaluated compared with pattern matching or statistical approaches, which cannot be selective in the choice of features relevant to an individual case. Another important advantage comes from the natural way that 'scale' can be accommodated. The boundaries usually occupy a region which is about 5x5 pixels, but in some cases it can be much larger, especially if texture is involved. The decision tree of rules can vary its region of analysis according to the local data, and appropriately classify an edge, say, having a large uniform region around. This avoids the complexities of evaluating at a variety of differing scales, a method advocated by other workers.

The approach which has been adopted here starts by evaluating a small number of measures around the centre of a region of interest, and these are used to select appropriate features to identify possible boundary types and orientations. Rules examine these features and hypothesise the existence of a boundary type, which if doubt remains is confirmed by suitable additional measures. In this way the system searches a tree of options until a conclusion is confirmed. The nature of this tree is such that this decision is usually reached very quickly.

The present implementation expresses the rules as simple conditionals in POP. This leads to a fast run speed but is cumbersome to modify. A more general (but slower) form of production system is being written which will allow rules to be applied in a more transparent fashion and will contain the basis for automatic addition of rules for a learning system. This production system could then be used to generate the simple conditionals to regain the required speed.

The system currently uses about 50 rules for the classification of boundary types and about 10 for the extension of a line into a new region. The basic structure of these rules is shown in Figure 2.

After finding the boundaries in the image, stored as a linked list of details about each point, a database of boundary segments is produced for the subsequent analysis. The present implementation can only extract straight line segments, but extension to arc segments is not difficult. Straight line segments are found by looking along the boundaries for bends or other features which denote a meaningful place to form a straight line end. These features include sharp changes in local orientation, excessive deviation from the previous path of a line, or proximity to the end of another line segment. The boundaries identified as straight lines in this way are fitted by least squares and the endpoints, length, mean contrast and mean brightness, orientation, length, and line type are formed into PROLOG facts, shown in Figures 3 and 4. This list of facts forms the starting database which is consulted by the shape finding expert described next.

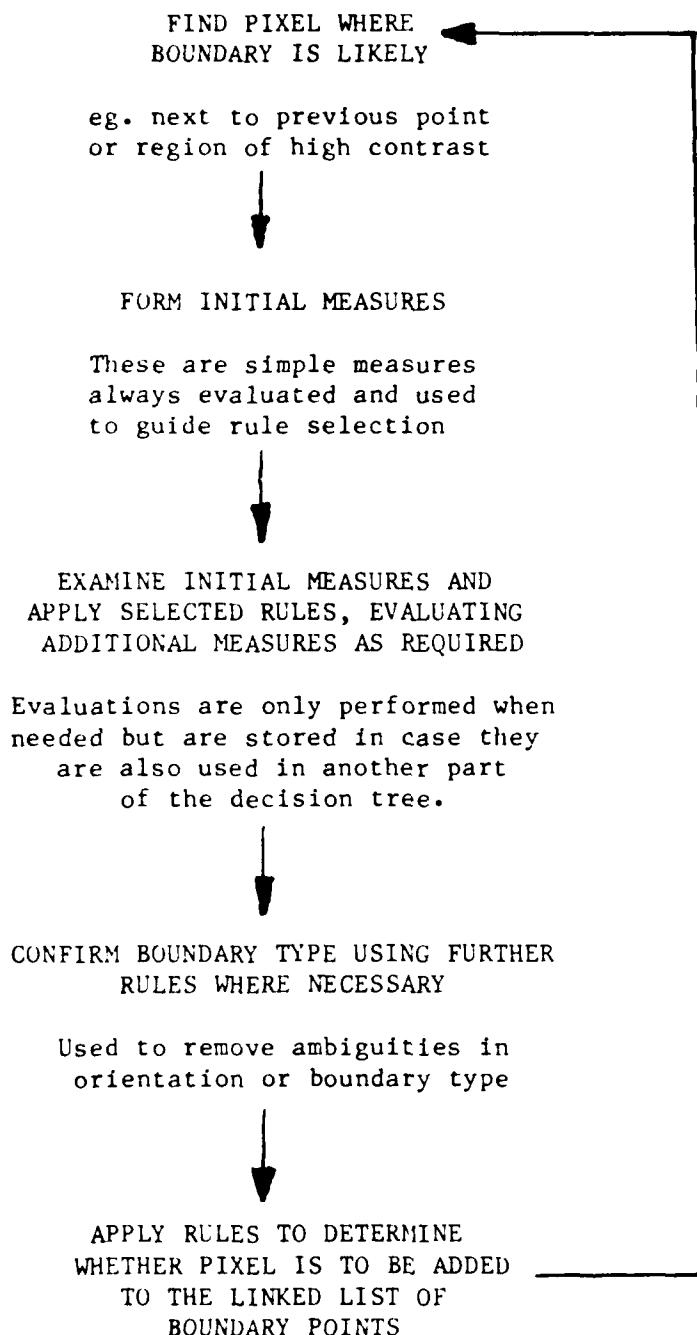


Figure 2. Summary of the Way Rules are Used to Find Boundaries in Images

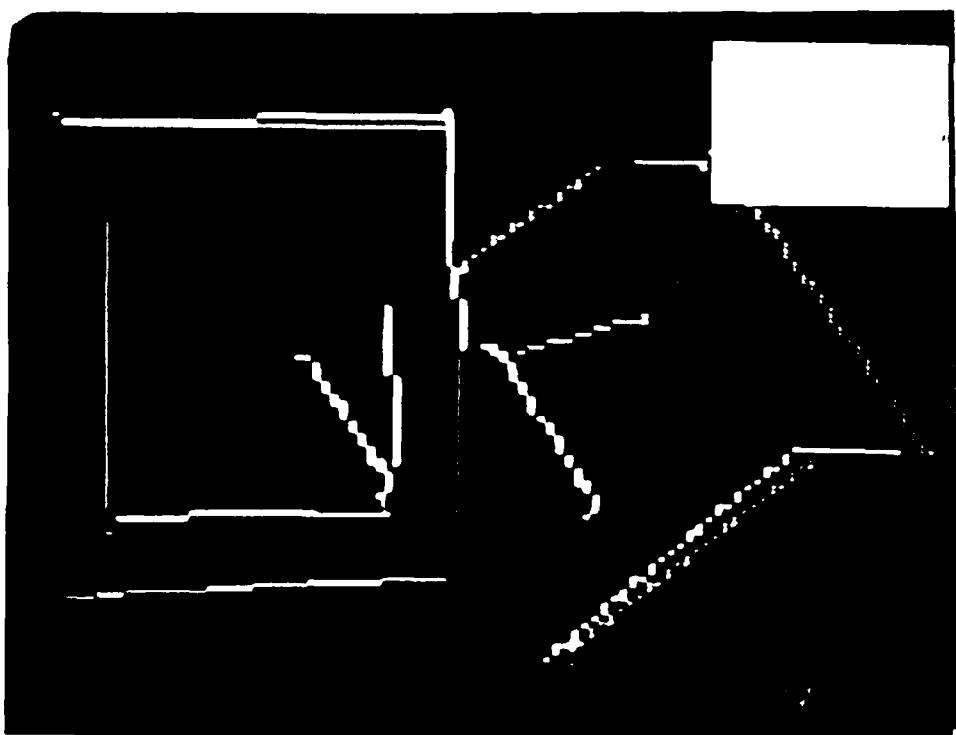


Figure 3(a). Boundary finding in hand drawn image.



Figure 3(b). Boundary finding in image of video cassette.

```

/* parameters are: */

/* (line no., real ends, fitted ends, strength, angle, length) */

line_seq(1,[65,59],[66,93], [65,59],[65,93],16,180,34).
line_seq(2,[64,92],[63,60], [64,92],[64,60],18,0,32).
line_seq(3,[79,30],[65,45], [79,30],[65,45],21,223,21).
line_seq(4,[77,107],[87,106], [77,106],[87,106],16,90,10).
line_seq(5,[57,76],[57,83], [57,76],[57,83],18,180,7).
line_seq(6,[56,51],[56,36], [56,51],[56,36],18,0,15).
line_seq(7,[78,84],[67,58], [79,84],[67,58],22,335,29).
line_seq(8,[79,78],[85,53], [79,78],[85,53],21,13,26).
line_seq(9,[50,63],[55,75], [50,63],[56,75],23,153,13).
line_seq(10,[102,75],[77,106], [102,75],[76,105],21,221,40).
line_seq(11,[73,109],[66,95], [74,109],[66,95],23,330,16).
line_seq(12,[87,108],[75,109], [87,108],[75,109],17,265,12).
line_seq(13,[56,75],[56,55], [56,75],[56,55],16,0,20).
line_seq(14,[101,73],[111,73], [101,73],[111,73],18,90,10).
line_seq(15,[110,75],[104,75], [110,75],[104,75],17,270,6).
line_seq(16,[113,76],[89,108], [113,76],[89,108],18,217,40).
line_seq(17,[91,27],[114,75], [91,27],[115,75],18,153,54).
line_seq(18,[80,27],[90,27], [80,27],[90,27],17,90,10).
line_seq(19,[65,41],[79,27], [65,41],[79,27],21,45,20).
line_seq(20,[64,22],[64,41], [64,22],[64,41],16,180,19).
line_seq(21,[21,22],[63,21], [21,23],[63,22],13,89,42).
line_seq(22,[18,97],[20,23], [19,97],[19,23],16,0,74).
line_seq(23,[65,94],[19,98], [65,94],[19,98],18,265,46).
line_seq(24,[63,22],[21,25], [63,24],[21,25],14,269,42).
line_seq(25,[21,26],[20,95], [21,26],[20,95],13,181,69).
line_seq(26,[21,95],[63,93], [21,96],[63,93],15,86,42).
line_seq(27,[67,93],[73,104], [67,93],[73,104],21,151,13).
line_seq(28,[74,105],[100,73], [74,105],[100,73],19,39,41).
line_seq(29,[99,72],[80,30], [99,72],[79,30],23,335,47).
line_seq(30,[82,30],[100,72], [81,30],[101,72],21,155,47).
line_seq(31,[90,30],[83,29], [90,29],[83,29],16,270,7).
line_seq(32,[111,71],[91,31], [111,71],[91,31],19,333,45).
line_seq(33,[111,76],[112,72], [111,76],[112,72],13,14,4).
line_seq(34,[88,105],[110,77], [88,105],[110,77],18,38,36).
line_seq(35,[57,52],[58,76], [57,52],[58,76],10,178,24).
line_seq(36,[26,35],[56,33], [26,36],[56,33],14,84,30).
line_seq(37,[26,87],[26,36], [26,87],[26,36],13,0,51).
line_seq(38,[57,84],[27,87], [57,84],[27,86],16,266,30).
line_seq(39,[55,36],[28,38], [55,36],[28,38],15,266,27).
line_seq(40,[27,39],[27,83], [27,39],[27,83],15,180,44).
line_seq(41,[28,84],[56,82], [28,84],[56,82],14,86,28).
line_seq(42,[56,81],[47,60], [57,81],[47,60],24,335,23).
line_seq(43,[48,59],[55,52], [48,59],[55,52],20,45,10).
line_seq(44,[70,59],[78,78], [69,59],[78,78],21,155,21).
line_seq(45,[84,53],[71,59], [84,54],[71,59],19,249,14).
line_seq(46,[67,57],[84,52], [67,57],[84,52],19,74,18).
line_seq(47,[86,53],[79,83], [87,53],[80,83],21,193,31).
line_seq(48,[56,54],[50,62], [56,54],[50,62],21,217,10).
line_seq(49,[65,46],[65,58], [64,46],[65,58],13,175,12).
line_seq(50,[63,59],[63,46], [63,59],[63,46],18,0,13).
line_seq(51,[57,33],[57,50], [57,33],[57,50],15,180,17).
line_seq(52,[63,45],[63,23], [63,45],[63,23],19,0,22).

```

Figure 4. Line segment database for Figure 3(a).

onto three parallelograms. An assumption that boxes will be seen in this way can be justified from a most general viewpoint argument. If the solid angle over which the projection will consist of only one or two rectangles is considered, it is seen to be small in comparison.

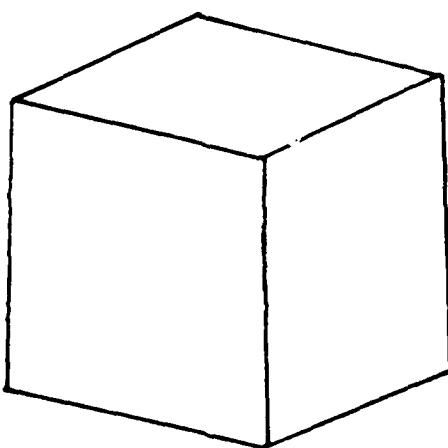
For particular images of a box, in view of the uncertainties involved in joining lines and measuring their orientations, it is inevitable that lower level processing will describe some parallelograms as rectangles. This will occur, of course, only when they are close to rectangular. This type of effect must be catered for by a level of tolerance in the processing; this will be realized in the form of flexibility of description at the higher level. So, to find a box in the image, the knowledge base could contain a description of a box in terms of three connected parallelograms and also a description in terms of two butted parallelograms and a rectangle. Alternatively, it could contain the relationship between rectangles and parallelograms, namely that the former is a specific version of the latter. For the rest of the discussion, the ability to reason that a rectangle is a parallelogram will be assumed.

The requirement that the constituent parts of the object connect in a certain way, is described in terms of the way common vertices are arranged in the image. A specific algorithm was developed to check these requirements in a meaningful order. As a precursor to this, the corners of the three shapes under study were converted back to the internal representation derived for the sketch form of the image. This allows a symbolic matching of coordinates to be performed.

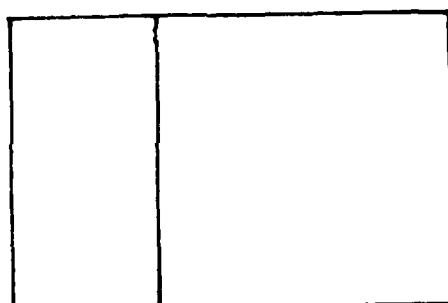
Three parallelograms are selected from the shape database. In principle, these could be joined by considering all possible orientations and any arrangement which matched all of the connectivity requirements would be asserted as evidence for a box in the image. If the match is unsatisfied, then one of the three parallelograms can be rejected, and a new one considered. Again, all orientations would be tried for a match. As usual with such a backtracking strategy, if no box can be found with the last parallelogram replaced, alternatives for the second parallelogram can be tried. Finally, if this is to no avail, alternatives for the first parallelogram can be considered. However, a general matching approach is not particularly efficient. By ordering the way in which the important vertices are found the procedure can be enhanced. First a candidate is found for the central vertex (Figure 13(a)). Next, the three possible outer triple vertices are considered, and finally the remaining corners. A representation of a shape's corner coordinates as a list structure allows the program to 'peg down' the centre and move along the list to access the adjoining vertices.

By finding the only possible centre, the number of combinations and orientations is vastly reduced. The matching of three parallelograms can then be terminated if they fail to satisfy rules describing the triple corners, or if some of these turn out to be identical. If these vertices can be found they completely define an instance of a box. It only remains to check that the remaining corners are non-coincident before a cube is identified. Before asserting it into the object database, any previously found cubes in the database are examined. In this way, duplication is avoided.

a)



b)



c)

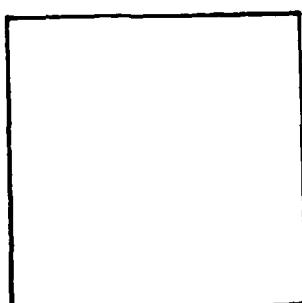


Figure 12. Possible views of a rectangular box.

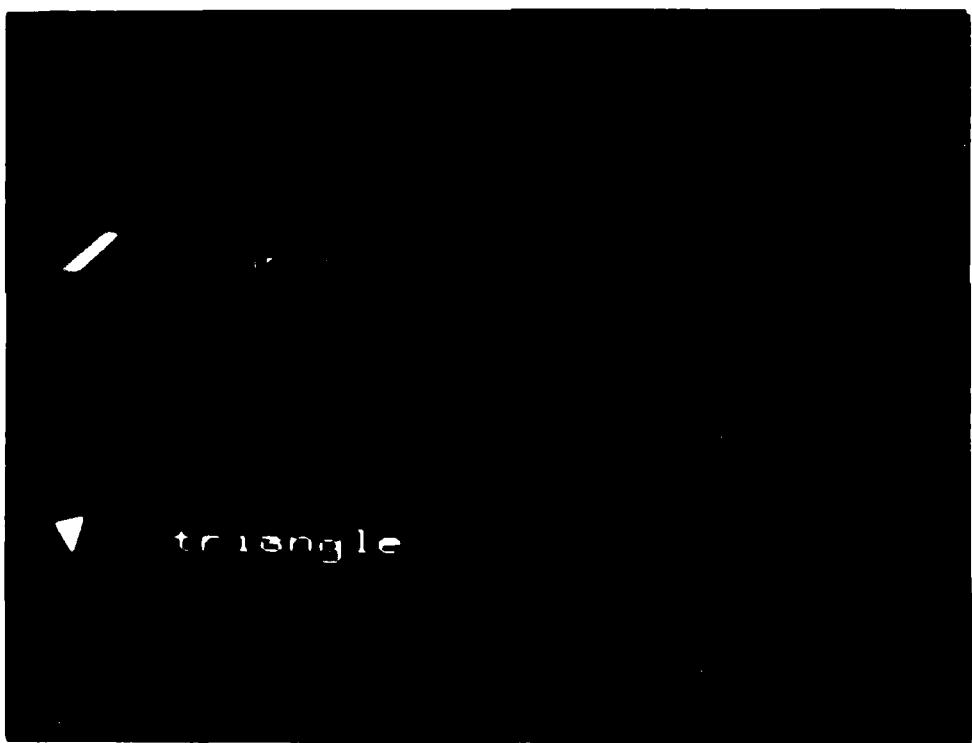


Figure 11. Interpretation of image including obscured rectangle.

concentrates on how partially obscured shapes might be proposed, and how supportive evidence may be used to corroborate these hypotheses. A specific example of the use of these ideas from the current work is given. For the purpose of this discussion the image of Figure 3(a) will be used. It contains a partially obscured rectangle, which is split into a pentagon and a triangle. The pentagon will be used to suppose a rectangle, and this will be supported by the existence of the triangle. Currently the software suite only deals with rectangles obscured in such a way as to produce a pentagon in the image, but this is readily extensible to cater for other obscurations as the method employed relies on evidence for the existence of a rectangle and not on the shape containing the evidence.

A pentagon produced by covering part of a rectangle will still display three right angles at consecutive corners. If these can be identified, then the underlying rectangle can be supposed and asserted. To do this the program makes a list of facts for each vertex in the shape under consideration. The elements of this list are the propositions 'true' and 'false', and the value of each element depend on whether the current vertex is, or is not, a right angle. The list is extended by adding the first two elements on to the end of the list; if this new list has three consecutive 'true' elements then the shape has the required form to suggest a hidden rectangle.

Two methods can be used to find the hidden corner of the rectangle. The coordinates of the three right angled corners of the shape are recalled, and are used to predict the fourth corner of the rectangle. Alternatively the line segments pointing toward the missing vertex can be continued, and used to predict the position. This is done using the procedure developed for finding the position of a known vertex from the lines forming it for shapes which are not obscured. The software is capable of using either method. Whichever is used, the suggested rectangle is asserted into the shape database, and then looked for.

To see if a suggested rectangle is further supported by the image database, a vertex at the hypothesised corner position is required. If the image database contains a vertex with the right properties, the existence of a rectangle is asserted into the shape database. An interpretation of the shape database after the obscured rectangle in Figure 3(a) has been identified is shown in Figure 11.

A natural extension of this idea is to incorporate further possibilities for the obscured shapes. The list based algorithm described above is able to cope with this generalization as the length of the list is not determined *a priori*; it only remains to describe more shapes in the knowledge base. Work with this aim is already underway.

2.2.5 Hypothesis of 3-D objects from shape database.

This section describes the method employed to suggest the existence of three dimensional objects in the image. The technique has been applied to the image under discussion, Figure 3(a).

Of the many possible images of a rectangular box, hinted at in Figure 12, by far the most likely is the view of Figure 12(a). The box is projected



Figure 10. Reconstruction of image from shapes found.

```
rectangle([27, 85], [57, 81], [56, 34], [27, 37]).  
rectangle([20, 96], [65, 93], [63, 22], [21, 24]).  
p_gram([88, 107], [111, 75], [102, 74], [76, 107]).  
p_gram([101, 73], [111, 74], [91, 29], [81, 29]).  
pentagon([65, 45], [66, 93], [74, 106], [100, 73], [80, 29]).  
triangle([56, 53], [56, 77], [49, 61]).  
triangle([79, 81], [69, 58], [85, 53]).
```

Figure 9. Database of shapes found.

shapes. The arc pairs are found and those which close on themselves are removed. The rest are asserted into the database. The same is done for arc triples; those in which an end point is degenerate with one of the interior point are removed. The concept of pairs and triples limits the shapes which can be considered easily to have less than six sides. However by allowing arc groups this can be extended.

The next step in the processing is to find the most general shapes in the image; namely the triangles and quadrilaterals. These are then made more specific by renaming them as parallelograms or rectangles. To find triangles, arc triples which form closed shapes are considered. Any for which two sides are collinear are retracted from the image database. The rest are certain triangles and are left subject to checking for degeneracy, in which case only one copy of the triangle is left. Quadrilaterals are formed from two arc pairs which join the same points and have distinct intermediate vertices. Again, collinearity of lines is checked and one copy of each well formed quadrilateral is kept. For pentagons an arc pair and an arc triple are joined, subject to all vertices being distinct.

To further classify shapes into rectangles, parallelograms and so forth, the line segments making up each shape found are checked. The angles of the various lines must be compared to see if they are parallel or lie at right angles. Rules containing a description of, for example, a rectangle as a particular type of quadrilateral are present in the analysis knowledge base. Having found these specific versions of shapes they are asserted into the image database and their less specific versions are removed. This is permissible as any higher level reasoning over shapes will know that a rectangle can also be viewed as a parallelogram and a quadrilateral.

Having found all simple shapes, these are put into a file of shapes for use in reconstructing the image. An example of such a file is shown in Figure 9, which has been generated from the image displayed in Figure 3(a). A philosophy adopted throughout this work has been that, at all stages of processing, the altered image should be available to the user so that, if required, a visual check can be made. The existence of this file does just this. It should be noticed that that file is quite small, and this reflects the data reduction which is achieved in performing the analysis.

The structure of the shape file is a set of PROLOG facts. These facts detail the type of each individual shape, as well as the coordinates of the vertices of the shape. These pieces of information are available to a POP-11 shape drawing routine. A reconstruction of the image at this stage is shown in Figure 10. The shape vertex coordinates are found by producing the lines from which the shape is constructed, and calculating the point at which the lines meet. This method will also be useful in joining otherwise unconnected shapes together as will be seen in the next section.

2.2.4 Hypothesising Partially Obscured Shapes.

In the previous section a description was given of how simple shape primitives could be found from the image database. This section

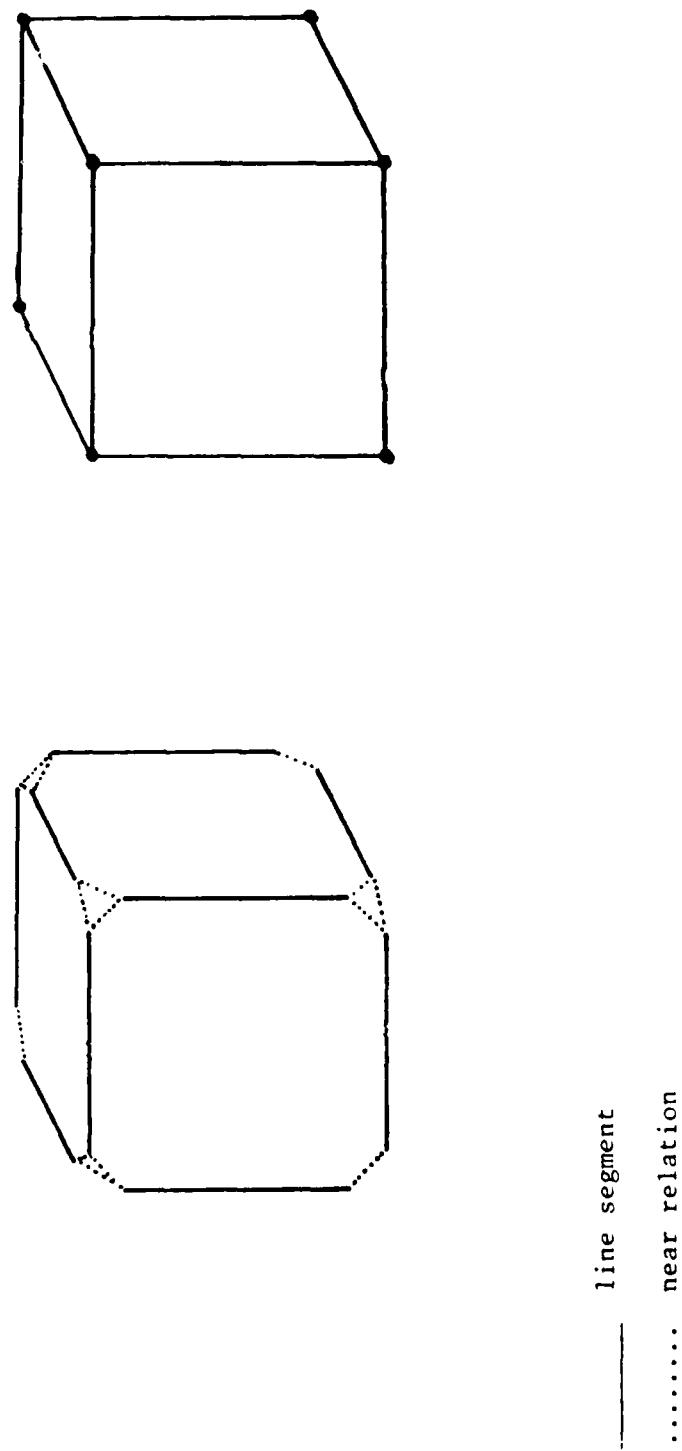


Figure 8. Collapse of nearness relations to symbolic nodes

c) Generating graph representation of image. The line segment and nearness relation databases can be used together to generate a simplified image database; the simplified database itself is used to steer the shape finding system. The image can be viewed as a graph, made from two distinct kinds of arcs. One set of arcs is simply the line segments themselves, and the other is the set of nearness relations which also link together the ends of line segments. The nearness relation arcs form clusters on the graph, which are themselves joined by the line segment arcs, as suggested in Figure 8(a). By collapsing these clusters down to single entities and remarking the relevant line ends as being at these nodes, the graph is simplified to a connected line sketch of the original image. The collapsed image line end groupings are assigned a symbolic label, which is carried forward into the rest of the processing; this makes the nodes look, in effect, like that shown in Figure 8 (b). It is interesting that a similar procedure could be used over the resulting graph representation of the image, using the line segments as the arcs, to extract unconnected objects in the scene. This would allow entirely separated parts of the image to be put into different databases and processed, in effect, as distinct images.

The algorithm used to separate the clusters can be viewed as follows. A node is chosen and marked with a star. All nodes joined to the marked node are then examined. If they are unmarked they are marked with a dagger and a star, if they are marked with a star they are left untouched. When all descendants of the chosen node are so dealt with, another node with a dagger is chosen. This dagger is removed and the process of considering descendants starts again. When no nodes are left still marked with a dagger, the entire of the node cluster has been found and are all marked with stars. These can then be collapsed and the process restarted with another unmarked node. When no nodes are left, then algorithm is terminated, and the image has been simplified. The newly found and collapsed nodes are then renamed and a new image database generated. The algorithm has been programmed in PROLOG; the sketch is, as usual, a set of PROLOG facts.

Having made explicit the relationships between the ends of lines, the collinearity of lines and having simplified the database to a sketch form, work can now start on the task of finding shapes in the database. This is the topic of the next section.

2.2.3 Shape Finding.

The method employed to find shapes is to locate closed paths within the image database. Of course, this method only accounts for complete, closed shapes in the image. For a variety of reasons, shapes may not be present with sufficient boundary contrast for them to be complete. As is discussed later, work is in hand to extend the technique to non-closed shapes, where lines have to be supposed.

As the main shapes of interest in this work are triangles, quadrilaterals and pentagons, a suitable algorithm for use is to first find all arc pairs and triples in the sketch graph, and then join these to form the required

Pixels

Figure 7. Distribution of distance between line ends.

accepts the line database as its input and creates an internal 'nearness' array. Any line ends which are close to each other cause the program to produce nearness relations, any which form parallel pairs are noted as such, and these are fed back into the image database in the form of PROLOG facts. This information was previously implicit in the database.

A closeness criterion is chosen to include all pairs of line ends which could meaningfully form part of a shape, but reject most combinations which could not. Looking at the statistics of the distance between line ends, several peaks are seen, as in figure 7. The peak at the lowest distance is due to the distribution of separations within the junctions in the image. The underlying more widely spread structure is due to the distances between line ends over the entire image. As the distance threshold is increased, up to a point, the processing includes more of the relevant near line ends. Eventually, if the distance threshold is too large, irrelevant connections are included. The boundary processing is assumed to be robust enough to be free from large gaps, and it has been found that a nearness criterion of about 6 pixels works well.

Work to improve this has already pointed the way to image dependent methods. Ideally, a distance criterion would be chosen so as to give a good interpretation of the image by capturing all meaningful shapes. Initially a small distance could be chosen, and this would be increased as the processing proceeded, and as the system felt necessary. By looping round under high level control, shapes and object would be found in an order reflecting the closeness of the vertices of which they are composed. Once a satisfactory interpretation had been reached, the looping would terminate. Alternatively, a choice for the distance threshold could be made based directly upon the statistics of the image content. From the above description, Figure 7, showing the distance statistics in the image, suggests that a threshold could be placed at the first dip in the distribution; this would separate out genuinely near line ends.

Further enhancement is possible if allowance is made for preferred directions at junctions; if two lines are virtually collinear then they are more likely to require joining. A further useful impact could be made by incorporating ideas from Gestalt visual psychology into the processing.

b) Rejoining separated line segments. In addition, long straight lines which have been broken by the lower level processing must be rejoined and asserted into the image database; the constituent parts are not removed as they may form part of an important shape in the image. This is performed using the nearness relations and the inter-line parallelism. Two lines which are parallel and join to each other are either a bar in the image, or are to be made into a longer line segment. In fact both cases lead to an extra entry in the image database. For the case of the longer line, the parallelism of the constituent lines to others is bestowed on the new line, and the nearness relations for the outer ends of the two lines are given to the required ends of the longer line.

```
/* A rectangle is defined
   as a closed shape made of right angles
*/
rectangle(Point1,Point2,Point3,Point4) :-  
    right_angle(Point1,Point2,Point3),  
    right_angle(Point2,Point3,Point4),  
    right_angle(Point3,Point4,Point5),  
    near(Point1,Point5).  
  
/* Definition of right angle as
   two lines meeting at 90 degrees
*/
right_angle(End1,Corner,End2) :-  
    line(End1,Corner,Parameters1),  
    line(Corner,End2,Parameters2),  
    angle(End1,Corner,End2,90).  
  
/* Line segment database
*/
line(start1,end1,parameters1).
line(start2,end2,parameters2).
line(start3,end3,parameters3).
```

Figure 6. Simple example of top-down PROLOG rules.

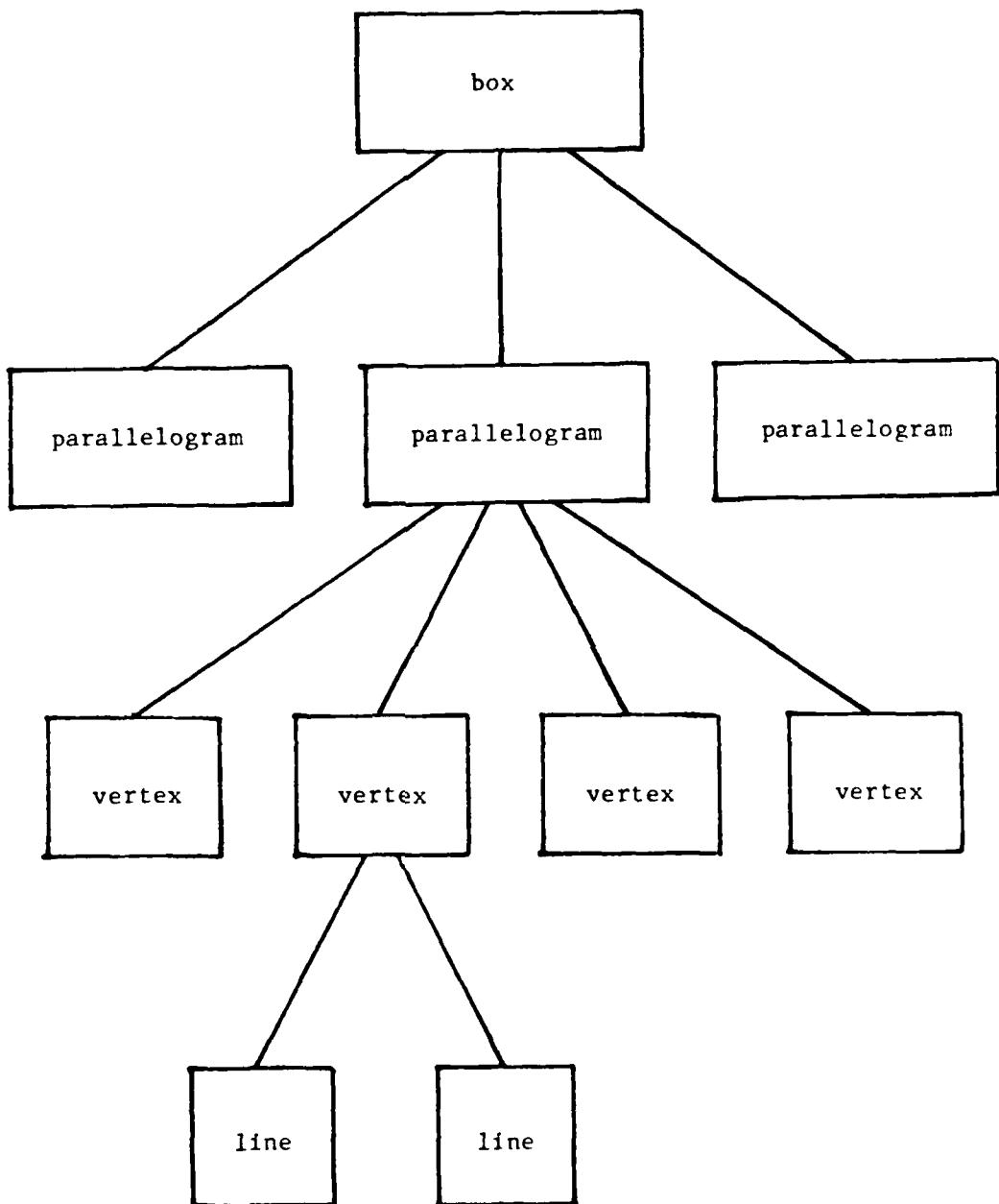


Figure 5. Example of representation hierarchy.

2.2 SHAPE AND OBJECT FINDING.

2.2.1 Approach to Shape and Object Finding.

A particular view of an object forms an image which is composed of several shapes; knowing the object these shapes can be predicted from the viewing direction. The shapes themselves are either general shapes like, for example, quadrilaterals or ellipses, or may be more specific, like squares or circles. They are, in turn, made from particular instances of vertices composed of lines meeting. This structural hierarchy is illustrated in Figure 5 for the image of a box.

The hierarchy immediately suggests the use of a top-down search in finding shapes and objects from a database of lines. Each object can be set as a goal to be satisfied by finding the shapes which compose it; the constituent shapes naturally give rise to further subgoals in terms of vertices and lines. This scheme is readily expressed in PROLOG, as suggested in figure 6 for a rectangle finder. This approach has been tried by the author [1], and very quickly leads to a combinatorial explosion for all but the simplest images. This happens because the same line segments are considered time after time in different contexts; they form parts which must be considered for all polygons, for example.

One way to overcome this difficulty to some extent is to use a bottom-up methodology. In such a system shapes are found by finding first all vertices present in the database, combining these to form closed shapes, and then restricting the shape categories to be more specific in nature. During the processing, therefore, extra knowledge of the image is generated as intermediate entities such as vertices are found and asserted; this knowledge is used in the search for other shapes. At any stage, most inference is performed on information from the preceding level of the structural hierarchy. This section of the paper describes how such a bottom-up approach has been successfully implemented using a suite of PROLOG rule bases. Note that the current work described here is concerned solely with shapes made from straight line segments.

2.2.2 Database Restructuring.

The database of line segments presented to the suite of software is in the form of PROLOG facts such as the one shown in Figure 4. Each fact details the start and end coordinates of the line, the orientation of the line, as well as other parameters such as the line strength, length and type. This database forms the only input to the software suite from the boundary finding processing of the image. It is therefore the initial database from which all reasoning about image content will be performed.

As the majority of inference of shape and object content of the image will require information about the interconnection and collinearity of these line segments, the data is restructured into a sketch form; close line ends are notionally joined to form a more simple view of the image. This is done in three stages:

- a) Finding nearness relations. This is done by a PASCAL program which

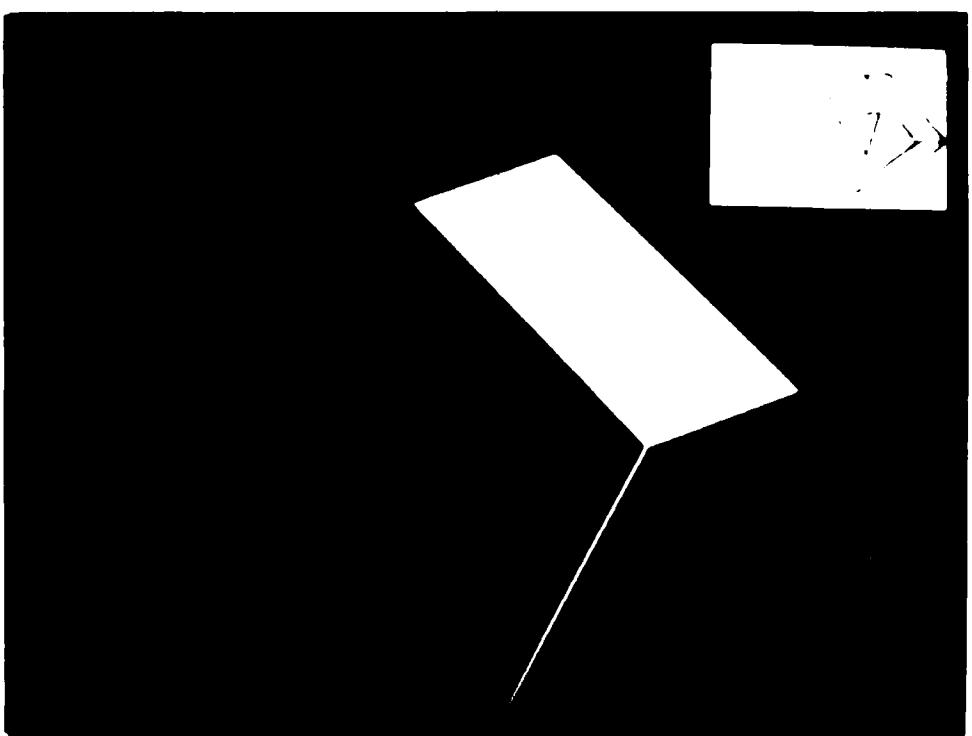
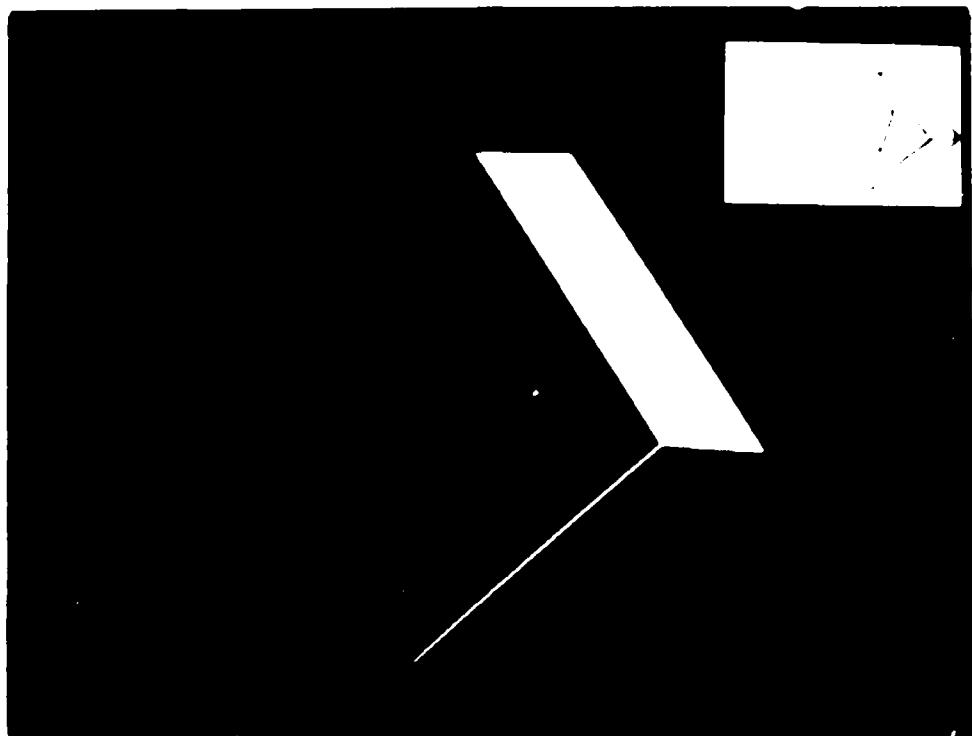


Figure 13. a) Box found in image.
b) Box rotated.

As with found shapes, the corner labeling of the box is converted back to image coordinates from its symbolic representation. Again this allows the object to be displayed to the user for verification. Figure 13(a) shows the visual output after the box has been found in the image under consideration.

The box that has been supposed may, of course, be the image of a peculiar skewed box-like object. Leaving this debate aside, and assuming that the object is a rectangular box, as may be justifiable from a priori knowledge, say, on a production line, further details about the object may be calculated. For example, the orientation of the box, or the coordinates of the hidden corner, may be found from relatively simple geometry. To demonstrate this ability, the program has been extended to allow the object to be rotated and displayed. Typical views of the cube-like object in Figure 3(a) are shown in Figure 13.

2.2.6 Extension of the technique.

The above discussion assumed that all shapes present in the image database are complete; namely closed. Under certain lighting conditions this assumption may not be a good one. Lines may be missing in the image, and this may lead to incomplete shapes. The previously described processing is not able to handle this situation. Defects of this sort can be partly overcome by hypothesising extra lines to close shapes. Such lines would not be added in an ad hoc way, but would be added as seemed reasonable. For example, if a rectangular shape faded slowly into shadow, a U shape would appear in the image. This shape, although not closed, would give rise to an arc triple (see Section 2.2.3). A line could be added to the database, such that it joined the two free ends of the U together, and it would be labelled as 'unknown'. This addition would produce an extra quadrilateral, also marked as 'unknown', which could be analysed to suggest and tentatively assert a possible parallelogram or rectangle, depending on the arrangement of the firm lines and their orientations. Any quadrilateral which cannot be made more specific is liable to be meaningless and would be removed from the database together with the added extra line. It must be emphasised that this would be a rather uncommon necessity. The boundary finding processing currently finds almost all boundaries, and the extensive use of tentative assumptions must always be avoided.

The technique for finding three dimensional objects has been demonstrated in section 2.2.5 for rectangular boxes. To extend this to further types of object, rules could be added which describe the new objects in terms of their projections into two dimensions, namely detailing their properties under imaging. This process could be mechanized, as the image of a general object can be predicted from simple geometry. The machine could be given an internal model of the object, either as a complete three dimensional construction, or as a description of its constituent parts, and would use this model to derive views of the object.

By describing all possible images of an object in terms of shapes already known to the system, and possibly also ascribing each view with a value for the likelihood that this view will be seen, rules could be generated which allow that shape to be recognized. The likelihoods contained within

the rules could be used to order all ambiguous interpretations leaving any final decisions to some higher level processing, using contextual information, for example.

Alternatively, the machine could be shown views of an object, and these used to either build an internal model of the object, or to directly derive rules for the recognition of the object within a scene. This would require the system to have a rich variety of basic shape and object primitives, and the system would also have to ensure that the newly acquired object was in some way differentiated from other objects already known to the system.

3. CONCLUSION

This paper has described an image understanding system with several interesting aspects which distinguish it from methods based on pattern matching or statistical pattern recognition. These include the use of a hierarchy of models to overcome the combinatorial problems normally encountered in image analysis, each model using knowledge local to its layer in the hierarchy; the use of declarative or rule based expression of knowledge, including that concerned with the low level interpretation of the pixels; and the use of problem solving methods which, although specific and implemented in the structure of the programs, provides a general methodology for forming inference about images without an exhaustive search of the problem domain.

Of particular note is the way the image is converted into a PROLOG database, the ability to recognise partially obscured shapes, and the inference of three dimensional shapes from the relationship of their two dimensional projections.

The methodology has been demonstrated with a simple hand drawn sketch. Extension to more complex shapes and objects is achieved by extending the knowledge base defining the various models, an activity which is currently in progress for military vehicles and production line components.

[1] Hearn D.B. (1983)
'Finding rectangles in images using PROLOG rules'
GW Division Working Paper, RSRE, August 1983.

DOCUMENT CONTROL SHEET

Overall security classification of sheet **UNCLASSIFIED**

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	RSRE MEMO 3683		
5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location Royal Signals and Radar Establishment		
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location		
7. Title An IKBS Approach to Image Understanding			
7a. Title in Foreign Language (in the case of translations)			
7b. Presented at (for conference papers) Title, place and date of conference			
8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date pp. ref.
Sleigh, A C	Hearn D B		July '84
11. Contract Number	12. Period	13. Project	14. Other Reference
15. Distribution statement Unlimited			
Descriptors (or keywords)			
continue on separate piece of paper			
Abstract This memorandum describes a method for extracting 2-Dimensional and 3-Dimensional shapes from scenes made up entirely of straight line segments. A new approach to extracting boundaries is discussed, and a methodology for defining geometrical shapes in PROLOG illustrated. Emphasis is placed on the structuring and control strategy for efficient implementation, and the applicability of IKBS methods to this is demonstrated.			

END

FILMED

5-85

DTIC